CSE231

Operating Systems Assignment 1

Name: Dhvanil Sheth

Roll No: 2021040

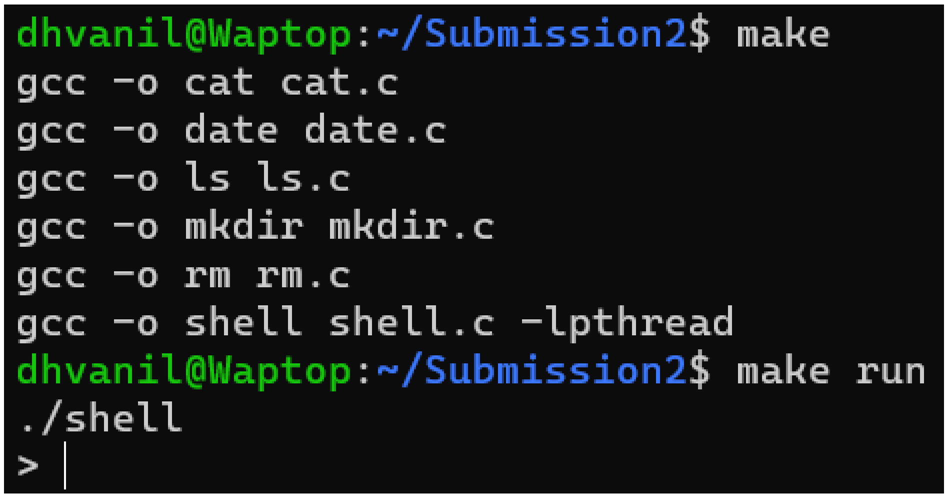
Section: A

Branch : CSE

I created this shell using C libraries and implemented the following three, internal commands – ‘cd’, ‘echo’ and ‘pwd’. These commands would be handled directly by the shell.
The shell also handles these 5 external commands - ‘ls’, ‘cat’, ‘date’, ‘rm’ and ‘mkdir’.

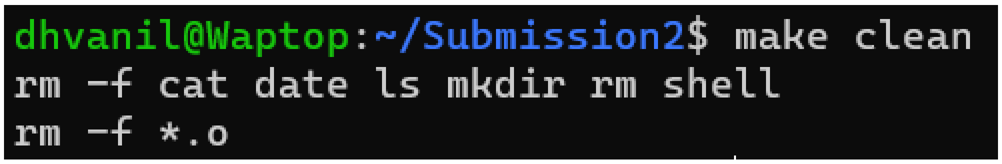
Makefile

Make compiles all the external as well as the shell containing the internal.
To run the shell “make run” is used



```
dhvanil@Waptop:~/Submission2$ make
gcc -o cat cat.c
gcc -o date date.c
gcc -o ls ls.c
gcc -o mkdir mkdir.c
gcc -o rm rm.c
gcc -o shell shell.c -lpthread
dhvanil@Waptop:~/Submission2$ make run
./shell
> 
```

“Make clean” clears out the object files as well as the executable files



```
dhvanil@Waptop:~/Submission2$ make clean
rm -f cat date ls mkdir rm shell
rm -f *.o
```

Internal Commands

Internal commands are interpreted by the shell program itself without requiring a different program to handle these operations.

1. cd is used to change the current working directory to a specified folder.

Options

● .. this command is used to move to the parent directory of the current directory, or the directory one level up from the current directory. “..” represents the parent directory.

● -P use the physical directory structure without following

Errors handled

● “no such file or directory”, when a user attempts to `cd` into a nonexistent entry.

● “invalid option”, when a user provides an invalid option.

● Throws perror when invalid input.

Assumptions made

● if no option is selected it goes to the home directory

● The user doesn’t enter cd - to go to the last directory instead uses cd ..

● When the user doesn’t enter cd with any directory it goes to home directory

2. echo displays a line of text in the terminal

Options

● -n do not output the trailing newline

● -e disable interpretation of backslash escapes

● -ne / en disable interpretation of backslash escapes

Errors handled

● “invalid option”, when a user provides an invalid option.

● Throws perror when invalid input.

Assumptions made

● If no option is selected it prints with a newline character

3. pwd prints the logical path of the current working directory, or physical .

Options

● -P/–physical avoid all symlinks

● -L/–logical resolve symbolic links (default)

Errors handled

● “invalid option”, when a user provides an invalid option.

● Throws perror when invalid input.

Assumptions made

● If no option is selected it prints the current working directory

External Commands

External commands are commands that aren’t implemented by the shell itself. The shell looks for their implementation and runs them as new child process

1. ls List files and directories present in either the working directory

Options

● -1 use a long listing format

● -a, --all do not ignore entries starting with . or ..

● -i, --inode print the index number of each file

Errors handled

● “invalid option”, when a user provides an invalid option.

● “no such file or directory”, when a user attempts to `ls` without a valid directory

● “not enough arguments” if the argument count is not as that required

Assumptions made

- If no option is selected it lists the files / directories without . or ..

2. date can print the system date and time.

Options

- -u, --utc print Coordinated Universal Time (UTC)
- -I, --iso-8601 output date/time in ISO 8601 format.

Errors handled

- “invalid option”, when a user provides an invalid option.
- “extra operand”, when a user provides extra unnecessary operands.

Assumptions made

- If no option is selected it prints the current local time

3. cat command concatenates file and prints on the standard output

Options

- -b, --number-nonblank number non empty output lines
- -n, --number number all output lines
- >> It reads data from the file and gives their content as output
- > Replaces file with information from new file

Errors handled

- “file or directory doesn't exist” when the file/directory doesn’t exist.
- “invalid option”, when a user provides an invalid option.
- “too many arguments ”, when a user provides extra unnecessary operands.
- “not enough arguments” when a user provides insufficient operands.

Assumptions made

- error is thrown when no input is given apart from cat

4. mkdir creates directories from those supplied in the arguments.

Options

● -p, --parents no error if existing, make parent directories as needed

● -v, --verbose print a message for each created directory

Errors handled

● “directory doesn't exist” when the file/directory doesn’t exist.

● “invalid option”, when a user provides an invalid option.

● “too many arguments ”, when a user provides extra unnecessary operands.

● “not enough arguments” when a user provides insufficient operands.

Assumptions made

● If no option is selected it tries to create a directory without verbose option

5. rm removes/deletes files that are supplied in the arguments.

Options

● -f, --force ignore nonexistent files and arguments, never prompt

● -r, --recursive remove directories and their contents recursively

Errors handled

(self explanatory)

● “file cannot be removed”

● “file or directory doesn’t exist”

● “directory cannot be removed”

● “directory cannot be read”

● “directory cannot be opened”

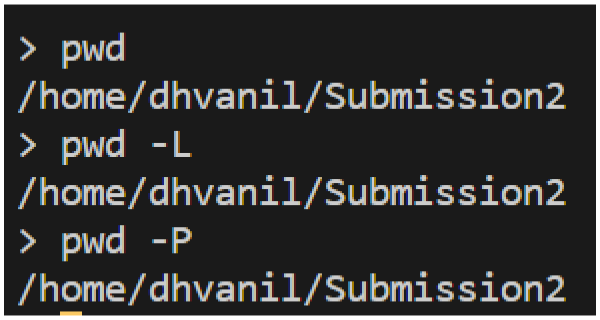
● “file cannot be read”

● “file can not be opened”

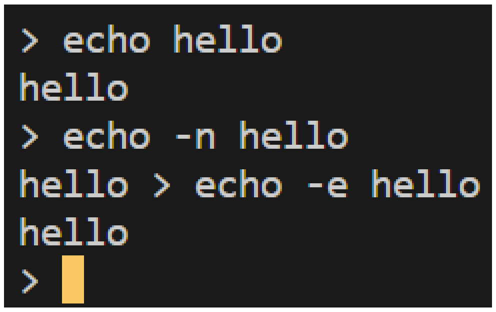
Assumptions made

● If no option is provided but arguments are provided then it attempts to deleted the given file/directory

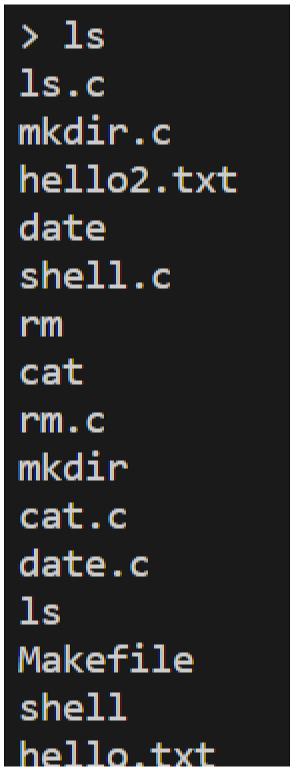
A few test cases



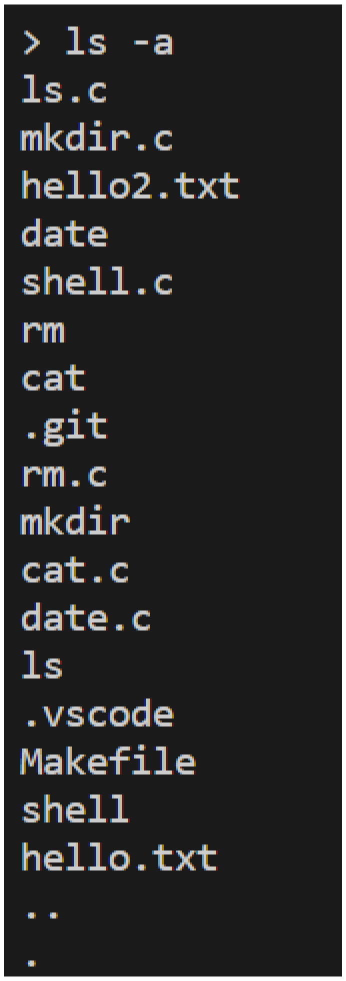
```
> pwd
/home/dhvanil/Submission2
> pwd -L
/home/dhvanil/Submission2
> pwd -P
/home/dhvanil/Submission2
```



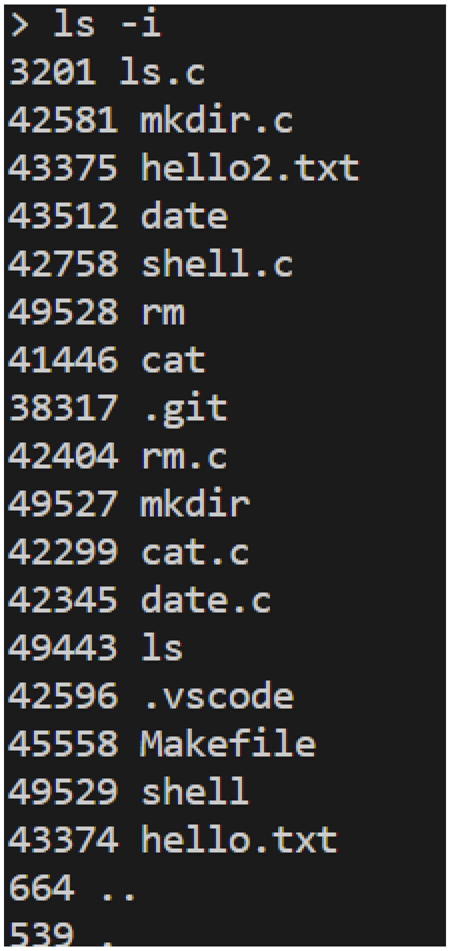
```
> echo hello
hello
> echo -n hello
hello > echo -e hello
hello
> 
```



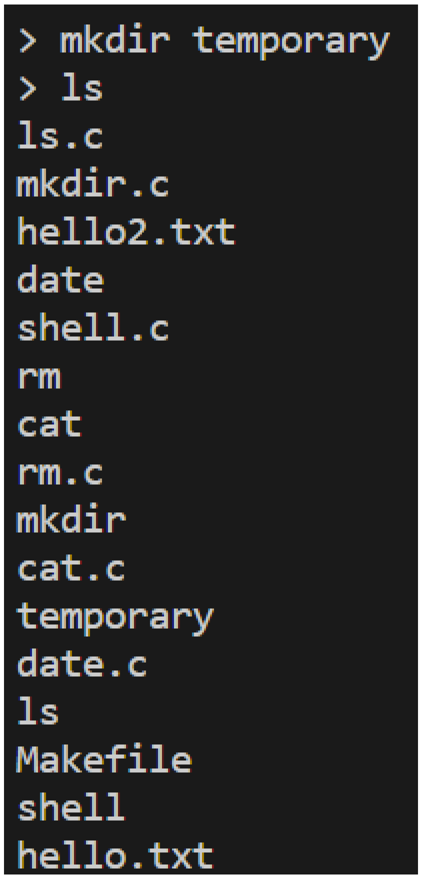
```
> ls
ls.c
mkdir.c
hello2.txt
date
shell.c
rm
cat
rm.c
mkdir
cat.c
date.c
ls
Makefile
shell
hello.txt
```



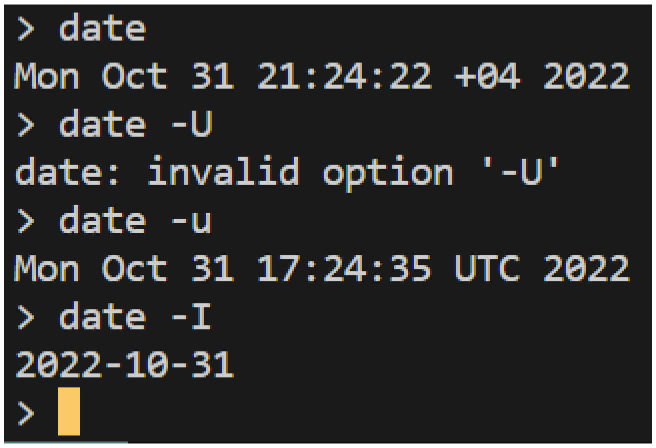
```
> ls -a
ls.c
mkdir.c
hello2.txt
date
shell.c
rm
cat
.git
rm.c
mkdir
cat.c
date.c
ls
.vscode
Makefile
shell
hello.txt
..
.
```



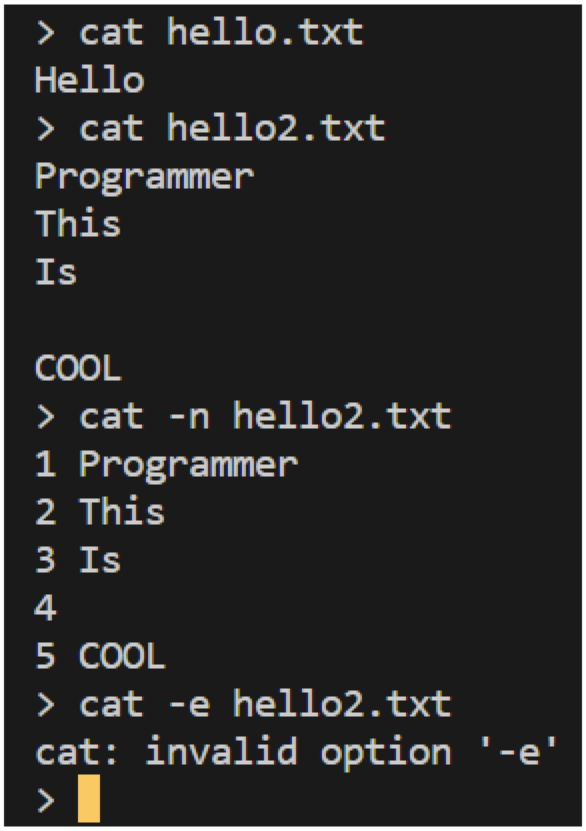
```
> ls -i
3201 ls.c
42581 mkdir.c
43375 hello2.txt
43512 date
42758 shell.c
49528 rm
41446 cat
38317 .git
42404 rm.c
49527 mkdir
42299 cat.c
42345 date.c
49443 ls
42596 .vscode
45558 Makefile
49529 shell
43374 hello.txt
664 ..
539 .
```



```
> mkdir temporary
> ls
ls.c
mkdir.c
hello2.txt
date
shell.c
rm
cat
rm.c
mkdir
cat.c
temporary
date.c
ls
Makefile
shell
hello.txt
```

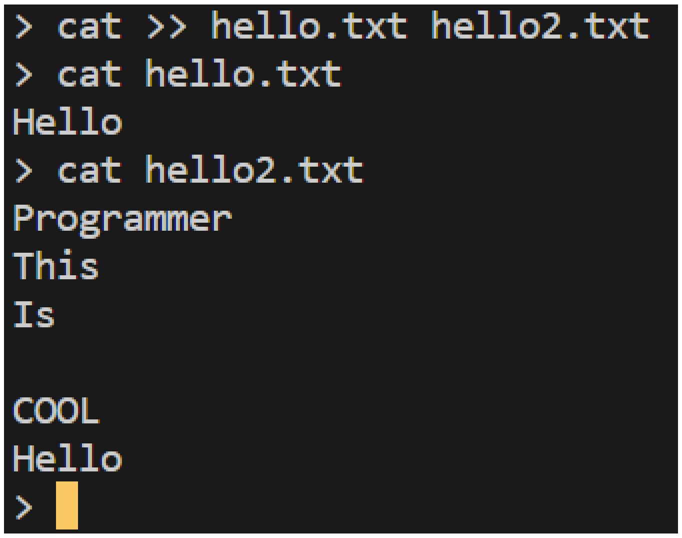


```
> date
Mon Oct 31 21:24:22 +04 2022
> date -U
date: invalid option '-U'
> date -u
Mon Oct 31 17:24:35 UTC 2022
> date -I
2022-10-31
> 
```



```
> cat hello.txt
Hello
> cat hello2.txt
Programmer
This
Is

COOL
> cat -n hello2.txt
1 Programmer
2 This
3 Is
4
5 COOL
> cat -e hello2.txt
cat: invalid option '-e'
> 
```



```
> cat >> hello.txt hello2.txt
> cat hello.txt
Hello
> cat hello2.txt
Programmer
This
Is

COOL
Hello
> 
```